Midterm Test #1 *(Sunday 16-11-2014)* **Time Allowed : 90 Minutes.**

***Answer all Questions*. TOTAL MARKS 20**

الإجابة النموذجية

Q1- (5pts) [20 minutes]

a) What is the impact of `jal` and `jr` MIPS instructions on the program counter?. ***(1 pt)*** **[3 min]**

Program Counter PC = Procedure address when executing `jal` instruction.
`jr rs :`
Program Counter PC = The contents of the register which is addressed by `rs` field.

b) How to push the contents of two MIPS registers into stack memory respectively?. ***(1 pt)*** **[3 min]**

For example we push `$s0` and `$s1` to the stack
1 : - Reserve two locations in the stack by using the following instruction

```
addi $sp,$sp,-8
```

2 : - By using store word instruction we can push `$s0`, and `$s1` to the stack memory as following :

```
sw $s0,0($sp)
sw $s1,4($sp)
```

c) How C compiler deals with overflow in addition and subtraction based on MIPS computers?. ***(1 pt)*** **[3 min]**

C compiler ignores overflow in addition and subtraction by using `addu, addiu,` and `subu` instructions.

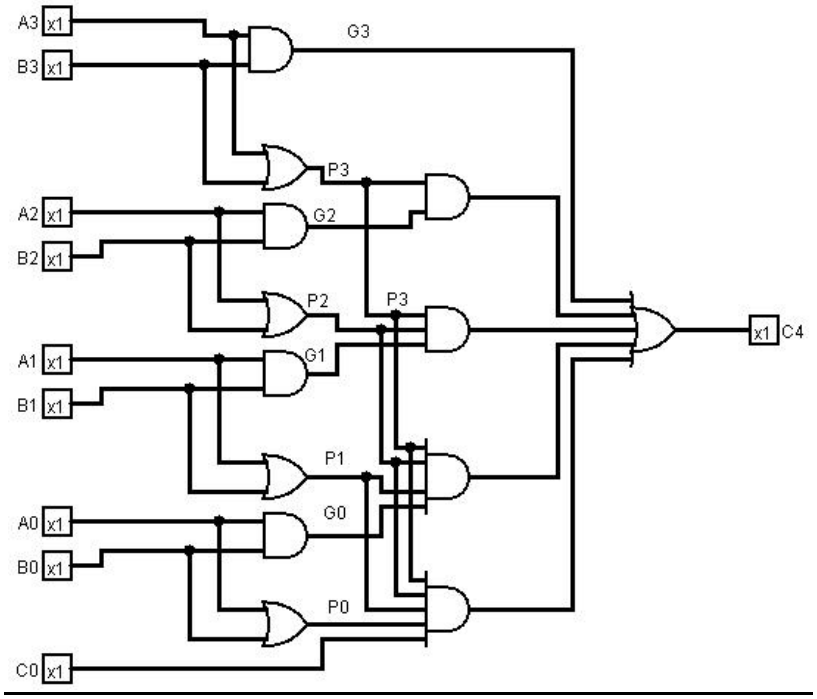
d) In details, how to improve the addition speed in a 4-bit ripple adder to become 3 gate delay?. ***(2 pt)*** **[11 min]**

To improve the addition speed in a 4-bit ripple adder to become 3 gate delay, we will use a carry-lock ahead adder (CLA).

In 1-bit full adder
Sum = A xor B xor Cin --> (1)
Cout = AB or ACin or BCin --> (2)
The gate delay in 1-bit full adder is 2 gate delay, in 4-bit ripple is 8 gate delay
The problem is in the carry out of each bit, we cannot get the sum of the second bit untill we get the value of Cout of the previous bit.
So let us do a calculaion to get Cout of the last bit in terms of Cout of the first bit as following : -
From eq (2) the carry will generated if A and B =1 --> G = A.B, and will propogated if Cin =1 and (A or B =1)--> P = (A+B). Cin,
So eq (2) could be written as
Cout = G + P.Cin
Carry out of the first bit
C1 = G0 + P0.C0
Carry out of the 2nd bit
C2 = G1 + P1.C1 --> C2 = G1 + P1.(G0 + P0.C0) --> C2 = G1 + P1.G0 + P1.P0.C0

Carry out of the 3rd bit
C3 = G2 + P2.C2 -->
C3 = G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.C0
Carry out of the 4th bit (Last Bit)
C4 = G3 + P3.C3 -->

C4 = G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.C0 --> (3)
From eq (3) we got C4 in terms of C0, by using the assumption in the textbook that each logic gate takes 1 gate delay we found that the gate delay in the longest path in CLA is only 3 gate delay as shown in the figure.



Q2- (8pts) [25 minutes]

a) Write a MIPS assembly program that performs the following processes :
(Assume all registers are initially zero)

i. Load a constant (0x45AC8B0) to $s0 without using `lui` instruction. (Three instructions) ***(1.5 pt)*** **[6 min]**
The constant is greater than 16 bit, without using `lui` instruction we can use shift instruction.

```
ori $s0,$s0,0x054A
sll $s0,$s0,16
ori $s0,$s0,0xC8B0
```

ii. Load A[2] to $t0, and A[3] to $t1 : A is a character array, and its base address is assigned to $s0. (Two instructions) ***(1 pt)*** **[3 min]**
A is a character array so its size is an unsigned 8-bit, as we mentioned in the class for C language.

```
lbu $t0,2($s0) # $t0 = A[2]
lbu $t1,3($s0) # $t1 = A[3]
```

iii. The element A[4] = A[2] if A[2] ≥ A[3], otherwise A[4] = A[3]. (Five instructions) ***(2.5 pts)*** **[11 min]**

```
      sltu $t2,$t1,$t0  #(is  $t1 < $t0 ? = is  A[3] < A[2] ? ) (= A[2]≥ A[3])
      beq  $t2,$0,else  # if $t2==$zero --> $t0 ≥ $t1 --> A[2] ≥ A[3] --> goto else
      sb   $t1,4($s0)   # A[4] = A[3]
      j exit # $t0 = A[2]
else: sb   $t0,4($s0)   # A[4] = A[2]
exit:
```

b) Calculate the average CPI and CPU cycles if the load/store instructions require 5 cycles to execute, branch/jump instructions require 3 cycles to execute, and the other instructions require 4 cycles to execute. ***(2 pts)*** **[5 min]**

The average CPI and CPU cycles are calculated if A[2] > = A[3] and if A[2]<A[3]
1: - if A[2]>= A[3]
Total IC = 8
Load/Store instructions = 3.
Branch/Jump instruction = 1
Other instructions = 4
Average CPI = (3*5+3*1+4*4) =34 clock cycles per instrcuction
CPU cycles = IC x CPI_{ave} = 8 * 34 = 272 cycles.

constant is greater than 16 bit, without using **`lui`** instruction we can use shift instruction.

```
ori $s0,$s0,0x054A
sll $s0,$s0,16
```

2: - if A[2] < A[3]
Total IC = 9
Load/Store instructions = 3.
Branch/Jump instruction = 2
Other instructions = 4
Average CPI = (3*5+3*2+4*4) =37 clock cycles per instrcuction
CPU cycles = IC x CPI_{ave} = 9 * 37 = 333 cycles.

Q3- (4pts) [15 minutes]

Perform the following multiplication as shown in the table below. Assume that the multiplicand register contains 101000 at the end of the multiplication process.

Iterations	1st Version unsigned multiplier			
	Steps	MR_{reg}	MC_{reg}	PR_{reg}
0	Initial Values			
1	1: Addition 2: Shift operations			
2	Shift operations			
3	1: Addition 2: Shift operations			
Stop	MR = _____, MC = _____, PR = _____			

MC reg = 101000 at the end of mutiplications means MC = 101 because at each iteration we shift the MC reg left 1-bit.In the steps column we find at step1 there is an addition ---> MR[0] =1, 2nd step M[0] =0, and the last step MR[0] = 1 ---> MR =101

Iterations	1st Version unsigned multiplier			
	Steps	MR_{reg}	MC_{reg}	PR_{reg}
0	Initial Values	101	000101	000000
1	1: Addition 2: Shift operations	101 010	000101 001010	000101
2	Shift operations	001	010100	000101
3	1: Addition 2: Shift operations	001 000	010100 101000	011001
Stop	MR = 5, MC = 5, PR = 25			

Q4- (3pts) [5 minutes]

Assume a 15 cm diameter wafer contains 84 dies, and has 0.020 defects/cm2. Assume a 20 cm diameter wafer contains 100 dies, and has 0.031 defects/cm2. Find the percentage of good dies from the total number of dies on each wafer. ***(2 pts)***. Which wafer has more number of good dies, and by how much. ***(1 pts)***

Load/Store instructions = 3

Yield = The percentage of good dies from the total number of dies on each wafer.

$$\text{Yield} = \frac{1}{(1+(\text{Defects per area} \times \text{Die area}/2))^2}$$

Wafer 1 : Die area = wafer aera / Dies per wafer

Wafer area = π r^2 = 22/7 * 7.5^2 = 176.79 cm^2

Die area = 176.79/84 = 2.11 cm^2

$$Yield = \frac{1}{\left(1+\left(0.02\times\frac{2.11}{2}\right)\right)^2} = 0.96 \rightarrow 96\%$$

Wafer 2 : Die area = wafer aera / Dies per wafer

Wafer area = π r^2 = 22/7 * 10^2 = 314.29 cm^2

Die area = 314.29/100 = 3.1429 cm^2

$$Yield = \frac{1}{\left(1+\left(0.031\times\frac{3.1429}{2}\right)\right)^2} = 0.91 \rightarrow 91\%$$

Wafer 1 good dies about 81 dies

Wafer 2 good dies about 91 dies

Wafer 2 has good number of good dies than wafer 1 by 1.123